Creating an entire online interview management system is a complex project that would require a team of developers, designers, and project managers. However, I can provide you with a basic outline of the components and some code snippets in Python and HTML to get you started. Keep in mind that this is a simplified example, and a production-ready system would require robust security, scalability, and many more features.

Here's an outline of the key components of an online interview management system:

1. \*\*User Authentication:\*\*

- Implement user authentication to allow candidates and interviewers to log in.

```python

# Python code for user authentication using Flask

from flask import Flask, request, session, redirect, url\_for, render\_template

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager, UserMixin, login\_user, login\_required, logout\_user, current\_user

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'your\_database\_url'

db = SQLAlchemy(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

password = db.Column(db.String(80), nullable=False)

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

user = User.query.filter\_by(username=username).first()

if user and user.password == password:

login\_user(user)

return redirect(url\_for('dashboard'))

return render\_template('login.html')

@app.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect(url\_for('login'))

```

2. \*\*Dashboard:\*\*

- After logging in, users should have access to a dashboard where they can view and manage their interviews.

```python

@app.route('/dashboard')

@login\_required

def dashboard():

# Retrieve interviews for the current user from the database

interviews = Interview.query.filter\_by(user\_id=current\_user.id).all()

return render\_template('dashboard.html', interviews=interviews)

```

3. \*\*Interview Creation:\*\*

- Allow interviewers to create new interviews, specifying details such as date, time, and location.

```python

@app.route('/create\_interview', methods=['GET', 'POST'])

@login\_required

def create\_interview():

if request.method == 'POST':

date = request.form['date']

time = request.form['time']

location = request.form['location']

# Create a new interview record in the database

interview = Interview(user\_id=current\_user.id, date=date, time=time, location=location)

db.session.add(interview)

db.session.commit()

return redirect(url\_for('dashboard'))

return render\_template('create\_interview.html')

```

4. \*\*Interview Display:\*\*

- Display the list of interviews and their details.

```html

<!-- HTML template for displaying interviews -->

{% for interview in interviews %}

<div class="interview">

<p>Date: {{ interview.date }}</p>

<p>Time: {{ interview.time }}</p>

<p>Location: {{ interview.location }}</p>

</div>

{% endfor %}

```

5. \*\*Database Model:\*\*

- Define the database model for interviews.

```python

class Interview(db.Model):

id = db.Column(db.Integer, primary\_key=True)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

date = db.Column(db.String(20), nullable=False)

time = db.Column(db.String(20), nullable=False)

location = db.Column(db.String(100), nullable=False)

```

This is just a basic starting point for an online interview management system. You would need to expand and refine this code, create appropriate HTML templates, add validation and error handling, and integrate other features like interview scheduling, candidate management, email notifications, and more. Additionally, ensure you follow best practices for security and data protection.